GNN-based Detection and Localization of IP VPN Misconfigurations

Authors: Fei-Fei Zhou

Date: 2024-01-07T00:00:00+00:00

Abstract

Network configuration verification, particularly for Virtual Private Networks (VPNs), is a complex task that must be performed prior to each production environment update to enable network providers to ensure network availability for their customers. This paper investigates a Graph Neural Network (GNN)-based approach for detecting and localizing configuration errors in IP Virtual Private Networks (VPNs). The research focuses on two distinct GNN models: one targeting routing configuration errors between customer and provider edge routers, and the other addressing VPN routing errors among different provider edge routers. The objective is to provide a tool that simplifies the process of end-to-end VPN configuration verification. In this study, a balanced dataset was employed to train both models, comprising examples of labeled configurations extracted from VPNs deployed on IMSNetworks. Results demonstrate that both models achieve high accuracy across VPNs of varying scales (ranging from 3 to 40 sites) and two architectural paradigms (full-mesh and hub-and-spoke). The principal advantage of this approach lies in the capability of Graph Neural Networks to capture complex relationships between network topology and configurations, thereby enabling more effective detection of configuration errors. Through this technique, network providers can validate network configurations prior to each update, thereby ensuring network availability for their customers.

Full Text

Preamble

Deep Learning Assignment: Detection and Location of Misconfigurations in GNN-Based IP VPN

Student Information:
Major & Class: Xinyan 2308

Name: Zhou Feifei
Student ID: 2023200853

Abstract: Configuration verification of networks, especially virtual private networks, is a complex task that must be performed before every production environment update to ensure network availability for customers. This paper explores a graph neural network (GNN)-based approach for detecting and locating configuration errors in IP Virtual Private Networks (VPNs). The study focuses on two GNN models: one targeting routing misconfigurations between customer and provider edge routers, and another addressing VPN routing errors between different provider edge routers. The goal is to provide a tool that simplifies the process of end-to-end VPN configuration verification.

In this research, both models were trained using a balanced dataset containing labeled configuration examples extracted from VPN deployments based on IMS Networks. The results demonstrate high accuracy for both models across VPNs of varying scales (from 3 to 40 sites) and two architectural types (full mesh and hub-and-spoke).

The advantage of this method lies in the GNN's ability to capture complex relationships between network topology and configuration, enabling more effective detection of configuration errors. By employing this technique, network providers can validate network configurations before each update to ensure customer network availability.

Keywords: Network Management, Configuration Error Detection, Deep Learning, Graph Neural Networks

---

Chapter 1 Introduction

1.1 Research Background and Significance

Layer 3 Virtual Private Networks represent the most common VPN type, typically referred to as IP VPNs. Operating at the network layer (Layer 3), they utilize the Internet to establish private communication channels between geographically dispersed networks. Both IPSec VPNs and SSL VPNs fall into this category.

These VPNs enable interconnection between different customer sites through an Internet Service Provider's (ISP) core network while ensuring quality of service and data flow isolation for each customer. Within the network, Provider Edge (PE) routers reside at the edge of the ISP's core network, with each PE router connecting to one or more Customer Edge (CE) routers representing different customer sites.

Providing VPN services based on Multiprotocol Label Switching (MPLS) infrastructure incurs substantial costs, primarily due to significant equipment and human resource investments. First, numerous routers and other network devices

must be purchased, as the service requires a robust network architecture for support. Second, professional network engineers must be hired to design and configure these devices, demanding considerable time and labor costs. Additionally, continuous maintenance and management are required to ensure network stability and security, representing a long-term, ongoing expense. Overall, this service is a high-cost, high-technical-skill offering typically suitable only for large enterprises or organizations with substantial requirements. General small-scale or individual users may find such costs prohibitive and unlikely to be necessary.

Configuring a service provider's backbone network is a critical and complex task requiring meticulous attention to detail. As the foundation for providing reliable and efficient connectivity to multiple customers, backbone configuration involves deploying a series of protocols across all backbone routers, including Interior Gateway Protocols (IGP), MPLS, and MP-BGP.

The complexity of the backbone configuration process demands precise protocol implementation to achieve seamless communication between routers. IGP propagates routing information within the service provider network to ensure optimal path selection. MPLS introduction enhances traffic engineering capabilities, while MP-BGP exchanges routing information between different Autonomous Systems (AS).

Despite this complexity, once configured, the backbone network tends to be relatively static, minimizing frequent modifications. In contrast, VPN Routing and Forwarding (VRF) tables and CE-PE routing configurations represent more dynamic and error-prone aspects of service provider operations. The continuous addition, modification, and deletion of customer sites introduces high variability, making this process susceptible to human error. The unique requirements of each customer site demand careful attention to detail, where simple oversights in parameter configuration can disrupt end-to-end connectivity between different sites.

As service provider networks expand to accommodate increasing numbers of customers and sites, the likelihood of configuration errors gradually rises. These errors may manifest as misconfigured VRF instances, incorrect CE-PE routing parameters, or inconsistencies in handling specific customer traffic. The consequences of such errors extend beyond individual customer sites, potentially affecting overall service quality and customer satisfaction.

[Figure 1: see original paper]-1 illustrates a simple example of BGP/MPLS IP VPN. Three customers are shown: Customer A has a full-mesh VPN interconnecting its three sites (A1, A2, and A3). Customer B has a hub-and-spoke VPN with specific routing policies allowing B2 and B3 to communicate only with B1. Finally, Customer C's VPN interconnects only two sites (C1 and C3).

1.2 Research Content and Article Structure

This paper investigates a Graph Neural Network (GNN)-based method for detecting and locating configuration errors in IP Virtual Private Networks (VPNs). Two GNN models are studied: one for routing misconfigurations between customer and provider edge routers, and another for VPN routing errors between different provider edge routers.

The paper is organized into five chapters:

Chapter 1: Introduction. This chapter presents the research background, significance, content, and organizational structure.

Chapter 2: Network Management and Graph Neural Networks. This chapter reviews the current state of network management research and provides an overview of graph neural networks.

Chapter 3: L3 VPN Configuration Data Model. This chapter details the two GNN models for VPN configuration, including the PE-PE routing model and CE-PE routing model.

Chapter 4: Training of Two Models. This chapter describes the training datasets and model architectures.

Chapter 5: GNN Model Performance Evaluation. This chapter presents experimental results and evaluation metrics.

Conclusion. This chapter summarizes the findings and contributions.

Chapter 2 Network Management and Graph Neural Networks

2.1 Current Research Status of Network Management

In recent years, network management based on machine learning has attracted significant attention [3], [4]. Reference [5] established a practical intelligent management system architecture for monitoring networks using CNNs of different depths, enabling model evaluation with real data from IP surveillance networks.

A dense Radio Access Network (dense-RAN) capable of radiation power management at base stations (BS) has been developed. Under constraints of user traffic and achievable rates, collaborative management of radiation power levels across multiple base stations can improve long-term network efficiency. To address the trade-off between complexity and performance, the authors used Deep Q-Networks (DQN) to model the overall optimization of multi-base station energy efficiency with multiplicative complexity constraints, achieving near-optimal performance [6].

Network traffic management represents a major challenge in modern Vehicular Cyber-Physical Systems (VCPS), as poor network resource management can degrade end-user Quality of Service (QoS). To address this issue, [7] proposed an SDN-enabled approach called SeDaTiVe, which uses a deep learning architecture to control incoming traffic in VCPS network environments. The advantage of using deep learning for network traffic control lies in its ability to learn hidden patterns in data packets and create optimal routing based on learned features. The deployment of smaller cells inevitably leads to more frequent handovers, making mobility management more challenging and reducing the capacity gains provided by dense network deployments. To fully capitalize on mobile user benefits in such network environments, [8] proposed an intelligent dual-connectivity mechanism for mobility management through deep learning-based mobility prediction. Most existing methods apply discretization to approximate continuous optimal values under actual driving conditions, resulting in relatively low performance and suffering from discretization errors and the curse of dimensionality.

Based on BP neural networks and the MTLBO algorithm, [7] established an MTLBO-BP neural network prediction model and tested its performance. The concept of modeling portfolio management through reinforcement learning is novel, and Deep Q-Networks have recently been successfully applied to portfolio management. A hierarchical Deep Q-Network-based framework was introduced, which addresses zero-commission problems by reducing the number of assets allocated to each Deep Q-Network and dividing the total portfolio value into smaller portions. Network slicing and Deep Reinforcement Learning (DRL) are important enablers for 5G and 6G networks, with methods for autonomously implementing each stage using Reinforcement Learning (RL) and DRL algorithms.

However, despite numerous successful applications of deep learning in network management, deep learning is rarely used for fault detection in networks, particularly for configuration error detection in IP VPNs.

2.2 Overview of Graph Neural Networks

Although traditional deep learning methods have achieved tremendous success in extracting features from Euclidean space data, many real-world application scenarios generate data from non-Euclidean spaces, where traditional deep learning methods still struggle to deliver satisfactory performance. For example, in e-commerce, a graph-based learning system can leverage interactions between users and products to make highly accurate recommendations, but the complexity of graphs poses significant challenges for existing deep learning algorithms.

This is because graphs are irregular—each graph has a variable number of unordered nodes, and each node has a different number of neighbors, causing some important operations (such as convolution) that are easily computed on images to become unsuitable for direct application to graphs. Furthermore, a core assumption of existing deep learning algorithms is that data samples are

independent of each other. However, this is not the case for graphs, where each data sample (node) has edges connecting it to other real data samples (nodes) in the graph, and this information can be used to capture interdependencies between instances.

In recent years, increasing interest has emerged in extending deep learning methods to graphs. Driven by successful developments in multiple fields, researchers have drawn inspiration from convolutional networks, recurrent networks, and deep autoencoders to define and design neural network architectures for processing graph data, giving rise to a new research hotspot—"Graph Neural Networks (GNN)."

To explicitly represent the relationships between different routers involved in VPNs, researchers have begun investigating Graph Neural Networks (GNN), a specialized neural network for graph-structured data that has recently received widespread attention [9], [10]. GNN models consist of multiple computational modules distributed across a set of layers. Graph Convolutional Networks (GCN) are among the most well-known computational modules for propagating information between nodes [11]. Each node in a graph embeds a set of features that are aggregated with its neighbors' features at each iteration. Other convolutional layers also exist, such as Edge-Conditioned Convolution (ECC) layers, where edge labels can be added to the graph to modulate information diffusion between neighbors [12]. Designing GNN models requires a series of steps: first specifying the structure and type of input graph, then defining the learning task. Depending on the problem formulation, various types of learning tasks are possible. Examples of learning tasks include node classification (e.g., learning new features for nodes) or graph classification (e.g., inferring properties of the graph itself), as shown in Figure 2 [Figure 2: see original paper]-2. In our context, modeling each VPN as a graph will enable us to capture the logical topology of the VPN, which directly impacts configuration parameters and their dependencies.

Furthermore, using graph data structures provides better expressive power than classical data structures (such as vectors or trees). Therefore, by relying on GNNs, we hope to address more complex networks and a greater variety of configuration errors than previous work [8].

Chapter 3 L3 VPN Configuration Data Model

3.1 Two GNN Models for VPN Configuration

When using Graph Neural Networks (GNNs) to solve VPN configuration problems, two main challenges are handling configuration parameters and policies across different sites and integrating customer subnets. This involves processing node features and managing interdependencies between configuration parameters.

First, regarding node feature processing, GNNs require all nodes to have the same number and type of features. Given that devices may have different configuration parameters, several approaches can address this issue. One method involves feature alignment, mapping devices with different configuration parameters to the same feature space through feature transformation networks or embedding layers. Another approach employs hierarchical models, designing independent GNN sub-models for each device type, allowing each sub-model to focus on configuration parameters specific to that device type.

Second, interdependencies between configuration parameters represent another important consideration. For graph structure design, appropriate graph structures can be used to represent dependencies between configuration parameters, such as using different types of edges to represent different dependency relationships. To better capture these relationships, subgraph-level attention mechanisms can be introduced, enabling the model to focus on processing configuration parameters for specific device types.

When configuring solutions for fault identification tasks, the overall task can be decomposed into multiple sub-tasks, with each sub-task focusing on specific types of devices or configuration parameters. This helps reduce the complexity of the learning task. Simultaneously, using multiple GNN models and fusing their outputs to obtain a comprehensive understanding of the overall configuration problem represents an effective strategy.

In summary, through methods such as feature alignment, hierarchical models, graph structure design, and task decomposition, GNNs can more effectively handle diverse configuration parameters and complex interdependencies in VPN configuration problems.

In GNNs, all nodes must have the same number and type of features, which is not the case for PE and CE devices with different configuration parameters. Another primary reason for selecting two different models relates to interdependencies between configuration parameters when identifying faults. Combining all configuration parameters for all fault types into a single model would again increase the complexity of the learning task. For example, faults affecting data forwarding between CE and PE routers depend only on the configuration of these two devices. Conversely, faults affecting data forwarding between specific customer sites depend on all VRFs deployed on PE routers involved in that customer's VPN.

3.2 PE-PE Routing Model

This model aims to detect and identify errors in VPN configuration using Graph Neural Networks (GNNs). In this model, each graph represents a VPN, with nodes representing different PE routers and edges representing connections between these routers.

The errors we focus on include failure to create VRFs on PE routers or incorrect

configuration parameters on VRFs, such as Route Distinguisher (RD) and Route Target (RT). Additionally, for non-full-mesh VPN setups, we also focus on errors in routing policy parameters.

Each node embeds information about the VPN configuration on that node, while binary features on graph edges indicate whether communication should occur between two nodes. This graph representation enables GNNs to learn relationships between nodes and topological information, thereby detecting and identifying configuration errors.

Overall, this approach improves the efficiency of detecting configuration errors by representing network configurations as graphs and leveraging the powerful capabilities of graph neural networks to process and understand complex network topologies and configuration relationships.

Through this model, our objective is to perform node classification. For each graph (i.e., each VPN), we provide an output for each node (i.e., each PE router) indicating whether configuration errors exist on that node. Since we want to pinpoint errors, each node's output is a vector of multiple classes, with each class representing a possible error. This learning approach is called supervised multi-class node classification. In our context, this type of classification enables detection and localization of PE routers containing VPN configuration errors for each VPN, precisely identifying them using the node's output vector.

Table 3 -1 lists the different errors we consider. Note that VRF, RD, and RT parameters must always be configured regardless of VPN type, while routing policy parameters are only required for non-full-mesh VPN setups.

Figure 3 [Figure 3: see original paper]-1 shows the structure of three graphs for three VPNs in the BGP/MPLS L3 VPN network topology example. Customer A has a full-mesh VPN, so binary features for all three edges are set to 1. Customer B has a hub-and-spoke topology, so one edge is set to 0. For Customer C, two edges are set to 1.

3.3 CE-PE Routing Model

This model's task is to detect and identify configuration errors that may affect data forwarding between CE and PE routers for each CE-PE connection. CE refers to Customer Edge, and PE refers to Provider Edge. Routing between these two components can be either static or rely on routing protocols. Table 3-2 enumerates all possible fault scenarios considered. These errors fall into four main categories: VRF (Virtual Routing and Forwarding), interface, eBGP, and static routing.

This model's approach represents each CE-PE connection as a separate graph, with each graph containing two nodes: one representing the CE router and the other representing the PE router. Each node's features specify the configuration parameters that can be set or modified when adding, updating, or deleting

VPN sites. Figure 3-2 shows the graph structure for CE-PE connections in the BGP/MPLS L3 VPN network topology example.

Unlike the previous model, this approach defines outputs for each graph rather than each node, implementing graph classification. This method reduces unnecessary complexity compared to having an output for each node. Therefore, the learning approach employs supervised multi-class graph classification. Each category represents a possible fault on the CE-PE connection, such as incorrectly configured IP addresses on interfaces or issues with eBGP sessions.

Figure 3-2 CE-PE Graph Structures for Three VPNs

The objective of this approach is to detect and identify configuration errors by learning graph representations of CE-PE connections. Through category labels in the output vector, the model can indicate the type of problem present on a specific connection, helping network administrators promptly correct configuration errors and improving overall network stability and maintainability.

Table 3-2 CE-PE Model Configuration Faults

Chapter 4 Training of Two Models

4.1 Training Set

To train and evaluate the PE-PE and CE-PE models, two datasets were constructed based on VPN configurations from IMS Networks customers [2]. IMS Networks has approximately hundreds of VPNs on its network, with each VPN comprising 10 to 30 remote sites distributed across 20 provider edge routers. Most of these VPNs are full-mesh, though certain VPNs have hub-and-spoke topologies requiring specific routing policies (to allow all sites in a VPN to communicate only with a unique central site called the hub). Since supervised machine learning methods require large amounts of labeled data, additional configuration examples had to be generated by creating random VPNs on the network backbone. To obtain a balanced dataset, 50% full-mesh VPNs and 50% hub-and-spoke VPNs were generated, which is necessary for the PE-PE model that configures routing policies. In the training set, simple errors were not merely injected; instead, random errors were added to simulate real-world scenarios.

Finally, 1000 IP VPN configurations of two types were obtained: full-mesh and hub-and-spoke, containing both correct and incorrect configurations. Each VPN has 10 to 30 CE routers distributed across 20 PE routers. These configurations were used to create two datasets for training and evaluating the PE-PE and CE-PE routing GNN models.

Consequently, the PE-PE routing dataset has 1000 graphs with 20 nodes each (one per VPN), with each node labeled according to its category. The CE-PE routing dataset contains over 20,000 labeled graphs, corresponding to the sum

of all CE-PE connections across all VPNs. For both models, 70% of the data was used for training, 10% for validation, and 20% for testing. Node categories in the above datasets correspond to the errors listed in both tables.

4.2 Two GNN Models

The Spektral framework was used to construct both GNN models: the PE-PE routing model and the CE-PE routing model. Both models employ the Keras API and TensorFlow 2 to provide a flexible framework.

The learning task for the PE-PE routing model is node classification. The model architecture includes two Edge-Conditioned Convolution (ECC) layers. The first ECC layer aggregates each node's features with its neighbors' features, while the second ECC layer serves as the output layer for classifying each node. ECC layers were chosen because edge features exist in PE-PE graphs that determine information communication between nodes, and ECC layers help process these edge features to achieve accurate node classification.

The CE-PE routing model's learning task is graph classification. The model architecture includes a Graph Convolutional Network (GCN) layer. Next, a pooling layer employs global sum pooling to extract high-level representations for each graph from node information. Finally, a Keras dense layer computes the output for each graph, completing the graph classification task.

Overall, both models utilize different convolutional and pooling layers within the Spektral framework to suit their respective learning tasks. The PE-PE routing model focuses on node classification, while the CE-PE routing model handles graph classification tasks.

Chapter 5 GNN Model Performance Evaluation

5.1 Parameters and Evaluation Metrics

The dataset comprises 300 graphs, each representing a VPN with 10 to 30 remote sites (i.e., CE routers) randomly distributed across 20 PE routers.

Precision and recall are calculated as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Where $Precision$ denotes precision, $Recall$ denotes recall, TP represents the number of correctly predicted positives, FP is the number of false positives

(other classes incorrectly predicted as this class), and FN is the number of false negatives (this class incorrectly predicted as other types).

The F1-score is used as the evaluation metric, representing the harmonic mean of precision and recall, with a maximum value of 1 and minimum of 0:

$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Using 80% of the PE-PE and CE-PE datasets for training and validation, the remaining 20% yielded a general F1-score (i.e., weighted average F1-score across all classes) of 94% for the PE-PE routing model and 98% for the CE-PE routing model.

5.2 Simulation Evaluation

5.2.1 PE-PE Model Figure 5 [Figure 5: see original paper]-1 plots the precision, recall, and F1-score values calculated using a dataset of 300 graphs, each representing a VPN with 10 to 30 CE routers randomly distributed across 20 PE routers. This provides 6000 PE nodes for classification. Classes 1 through 8 correspond to configuration errors listed in Table 3-1, while class 0 indicates no error on the node. Except for classes 5 and 7, which have F1-scores near 80%, all other F1-score values are close to 90%.

Classes 5 and 7 represent errors on subnets that should be configured on import and export routing policies, respectively. These two parameters are used in hub-and-spoke VPNs to filter routes imported into or exported from customer routing tables. The model's poor performance on these categories stems from two main reasons. First, the subnet parameter has no other features in the dataset for comparison or correlation (it only appears elsewhere on customer edge routers in the CE-PE model). Second, the training dataset is imbalanced for these error types. Although hub-and-spoke VPNs constitute 50% of the training dataset, routing policy errors are less representative than other errors that can occur on any VPN.

Classes 6 and 8 are also routing policy errors but focus on Route Target (RT) values (matched on import policies and set on export policies). Unlike classes 5 and 7 with subnets, these RT values correlate with RT values present in the customer VRF on the same PE node. The results demonstrate that the model can learn relationships between features representing VRFs and routing policy parameters.

The bar chart in Figure 5-2 confirms previous results. General precision, recall, and F1-score values were calculated by testing the PE-PE model on three different datasets: one consisting only of full-mesh VPNs, one only of hub-and-spoke VPNs, and one containing both types.

Figure 5-2 Detailed Performance by VPN Type

The chart clearly shows that the model learns better on full-mesh VPNs (i.e., without routing policies).

5.2.2 CE-PE Model Figure 5-3 CE-PE Performance per Fault Type

Figure 5-4 CE-PE Performance after Fault Classification

These tests were conducted on the same dataset used for the charts in Figure 5-1. Due to the many error types in this model (30 classes), for better readability we chose to group them by category.

Figure 5-5 F1-Scores for Different Sites in VPNs

Figure 5 shows F1-scores for PE-PE and CE-PE routing models using seven datasets, each containing 300 VPNs. We observe that changing the number of CEs per VPN does not affect the CE-PE routing model. This makes sense because changing the number of CEs does not alter anything on the CE-PE graph; it only changes the number of graphs to classify. For the PE-PE routing model, the global F1-scores for VPNs with 10, 20, and 30 CEs per VPN are similar to the average F1-scores plotted in Figure 3 (approximately 90%). This result is consistent because the model was trained on a dataset containing VPNs with 10 to 30 CE routers. More interesting results are the F1-scores for VPNs with 3, 5, 40, and 50 customer routers. Although these VPN sizes do not exist in the training dataset, classification accuracy remains excellent (F1-score only decreases by 3% or 4%), indicating that the learning process generalizes well without overfitting.

Conclusion

This paper investigated a graph neural network-based approach for detecting and locating configuration errors in IP VPNs. The research encompassed two distinct GNN models, each addressing different classification problems. The PE-PE model performs node classification to predict possible configuration errors on VRF tables or routing policies for each graph. The CE-PE model performs graph classification to predict possible configuration errors on customer-provider routing.

Testing on different datasets sorted by error type, VPN topology, and VPN size shows prediction accuracy for detecting configuration errors between 80% and 90%. Furthermore, tests on VPN patterns not present in the training dataset (i.e., with different numbers of sites and different locations on provider edge routers) show only a 4% performance decrease, indicating that the models perform well on unseen data.

References

[1] E. Rosen and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)," RFC Editor, RFC 4364, Feb. 2006.

[2] E.-H. Mohammedi, E. Lavinal and G. Fleury, "Detecting and locating configuration errors in IP VPNs with Graph Neural Networks," NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 2022, pp. 1-6.

[3] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, "Machine learning for networking: Workflow, advances and opportunities," IEEE Network, vol. 32, no. 2, p. 92–99, Mar. 2018.

[4] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, and et al., "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," Journal of Internet Services and Applications, vol. 9, no. 16, Jun. 2018.

[5] C.-C. Kao, Y.-C. Lai, J. Pei, C.-W. Chang, F.-H. Kuo and J.-Y. Shun, "Intelligent Management System: Deep Convolutional Neural Networks for Automatic Attribute Recognition in IP Surveillance Networks," 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS), Daegu, Korea (South), 2020, pp. 397-400.

[6] Y. Chang et al., "Collaborative Multi-BS Power Management for Dense Radio Access Network Using Deep Reinforcement Learning," in IEEE Transactions on Green Communications and Networking, vol. 7, no. 4, pp. 2104-2116, Dec. 2023.

[7] A. Jindal, G. S. Aujla, N. Kumar, R. Chaudhary, M. S. Obaidat and I. You, "SeDaTiVe: SDN-Enabled Deep Learning Architecture for Network Traffic Control in Vehicular Cyber-Physical Systems," in IEEE Network, vol. 32, no. 6, pp. 66-73, November/December 2018.

[8] C. Wang, Z. Zhao, Q. Sun and H. Zhang, "Deep Learning-Based Intelligent Dual Connectivity for Mobility Management in Dense Network," 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall), Chicago, IL, USA, 2018, pp. 1-5.

[9] Z. Liu and J. Zhou, "Introduction to graph neural networks," Synthesis Lectures on Artificial Intelligence and Machine Learning, vol. 14, no. 2, pp. 1–127.

[10] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," AI Open, vol. 1, pp. 57–81, 2020.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," CoRR, vol. abs/1609.02907, 2016.

[12] M. Simonovsky and N. Komodakis, “Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs,” in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

Note: Figure translations are in progress. See original paper for figures.